Contents & Goals

Last Lecture:
• Motivation: model-based development of things (houses, software) to cope with complexity, detect errors early
• Model-based (or driven) Software Engineering

This Lecture:
• Educational Objectives:
  • Why is UML of the form it is?
  • Shall one feel bad if not using all diagrams during software development?
  • What is a signature, an object, a system state, etc.? What’s the purpose of signature, object, etc. in the course?
  • How do Basic Object System Signatures relate to UML class diagrams?

Content:
• Brief history of UML
• Course map revisited
• Basic Object System Signature, Structure, and System State

Why (of all things) UML?

• Note: being a modelling language doesn’t mean being graphical (or: being a visual formalism [Harel]).
• For instance, [Kastens and Bünig, 2008] also name:
  • Sets, Relations, Functions
  • Terms and Algebras
  • Propositional and Predicate Logic
  • Graphs
  • XML Schema, Entity-Relation Diagrams, UML Class Diagrams
  • Finite Automata, Petri Nets, UML State Machines

• Pros: visual formalisms are found appealing and easy to grasp.
Yet they are not necessarily easy to write!
• Beware: you may meet people who dislike visual formalisms just for being graphical—maybe because it is easier to "trick" people with a meaningless picture than with a meaningless formula.

A Brief History of UML

• Boxes/lines and finite automata are used to visualise software for ages.
• 1970’s, Software Crisis™ — Idea: learn from engineering disciplines to handle growing complexity.
  • Languages: Flowcharts, Nassi-Shneiderman, Entity-Relation Diagrams
• Mid 1980’s: Statecharts [Harel, 1987], StateMate™ [Harel et al., 1990]
• Early 1990’s, advent of Object-Oriented - Analysis/Design/Programming
  • Inflation of notations and methods, most prominent:
    • Object-Modeling Technique (OMT) [Rumbaugh et al., 1990]
    • Booch Method and Notation [Booch, 1993]
A Brief History of UML

- Boxes/lines and finite automata are used to visualize software for ages.
- 1970s, Software Crisis™ - Idea: learn from engineering disciplines to handle growing complexity.
- Languages: Flowcharts, Nassi-Shneiderman, Entity-Relation Diagrams
- Mid 1980s Statecharts [Harel, 1987], StateMate™ [Harel et al., 1990]
- Early 1990s, advent of Object-Oriented Analysis/Design/Programming - Inflation of notations and methods, most prominent:
  - Object Modeling Technique (OMT) [Rumbaugh et al., 1990]
  - Booch Method and Notation [Booch, 1993]
  - Object-Oriented Software Engineering (OOSE) [Jacobson et al., 1992]
Each “persuasion” selling books, tools, seminars...
- Late 1990s: joint effort UML 0.x, 1.x Standards published by Object Management Group (OMG), “international, open membership, not-for-profit computer industry consortium”.
- Since 2005: UML 2.x

Common Expectations on UML

- Easily writeable, readable even by customers
- Powerful enough to bridge the gap between idea and implementation
- Means to tame complexity by separation of concerns (“views”)
- Unambiguous
- Standardized, exchangeable between modeling tools
- UML standards say how to develop software
- Using UML leads to better software

We will see...

Seriously: After the course, you should have an own opinion on each of these claims. In how far is what sense does it hold? Why? Why not? How can it be achieved? Which ones are really only hopes and expectations? ...?
Note: Object identities only have the "operation; analogously \( C(D) = C \).

Recall: by definition, seek a \( \tau \) from \( V \); attribute values \( \{C \} \) are mapped to an infinite set \( \{\} \). Object identities of different classes are disjoint, i.e.

\( \forall \) object identities of different classes are disjoint, i.e.

\( C(D) = C \).

Example:

Basic Object System Signature

Another Example

Common Semantical Domain
References


